Feuille de Travaux Pratiques sur les Matrices

Afin d'implémenter l'algèbre usuelle des matrices de nombres réels, on considère la structure C++ suivante.

```
struct Matrice{          // Matrice de réels
        int nl;          // nombre de lignes
        int nc;          // nombre de colonnes
        double** tab2d; // tableau de réels
};
```

1. Soient n, m deux entiers strictement positifs. Écrire une fonction

   ```
   Matrice zeros(int n, int m);
   ```

   construisant une matrice nulle de n lignes et m colonnes.

2. Soient x un nombre réel, i, j deux entiers positifs. Écrire une procédure

   ```
   void affecte_coeff(Matrice &M, int i, int j, double x);
   ```

   affectant le réel x au coefficient d'indices (i, j) d'une matrice M. On prendra en compte la précondition

   ```
   0 <= i < M.nl et 0 <= j < M.nc.
   ```

3. Soient i, j deux entiers positifs. Écrire une fonction

   ```
   double coeff(Matrice M, int i, int j);
   ```

   renvoyant la valeur du coefficient d'indices (i, j) d'une matrice M. On prendra à nouveau en compte la précondition

   ```
   0 <= i < M.nl et 0 <= j < M.nc.
   ```

4. Soit n un entier strictement positif. Écrire une fonction

   ```
   Matrice identite(int n);
   ```

   construisant la matrice identité de taille n.

5. Écrire une procédure

   ```
   void affiche(Matrice M);
   ```

   d'affichage d'une matrice.

6. Écrire une fonction

   ```
   Matrice somme(Matrice A, Matrice B);
   ```

   permettant de calculer la somme de deux matrices A et B. Les matrices A et B devront avoir les mêmes dimensions.

7. Écrire une fonction

   ```
   Matrice multiplication(Matrice A, double x);
   ```

   permettant de calculer le produit d'une matrice A par un réel x.

8. Écrire une fonction

```
Matrice produit(Matrice A, Matrice B);
```

permettant de calculer le produit de deux matrices A et B. Les matrices A et B devront avoir des dimensions compatibles.

9. Écrire une fonction

```
Matrice hasard(int n, int m);
```

construisant une matrice de dimensions (n, m) avec des coefficients aléatoires.

10. Écrire une procédure

```
void libere(Matrice &M);
```

de libération de mémoire.

11. Soit n un entier positif. On souhaite calculer rapidement la puissance n-ième d'une matrice carrée A.

(a) Écrire une fonction itérative

```
Matrice puissance(Matrice A, int n);.
```

(b) Écrire une fonction récursive

```
Matrice expo_rapide(Matrice A, int n);
```

à partir de l'algorithme suivant.

```
FONCTION exponentiation_rapide(A: Matrice, n: entier): Matrice
    Si n=1 alors
        renvoyer A
    Sinon
        Si n est pair alors
            renvoyer exponentiation_rapide(A^2, n/2)
        Sinon
            renvoyer A*exponentiation_rapide(A^2, (n-1)/2)
        Fin Si
    Fin Si
```

(c) Estimer les complexités temporelles des deux fonctions précédentes.

12. Implémenter l'algorithme de Strassen permettant un calcul rapide du produit de deux matrices.

13. Soit $P(X) = a_0 + a_1X + \cdots + a_nX^n$ un polynôme à coefficients réels. Implémenter une fonction permettant d'évaluer le polynôme $P(X)$ sur une matrice A.